DETEKSI MATA MENGANTUK DENGAN EFFICIENTNETV2 + ATTENTION DAN DUAL-ATTENTION

Syauqi (215150307111010)
Nandito Yuda Samosir (215150301111014)

METODE

Deteksi Mata Mengantuk menggunakan model

EfficientNetV2 + Attention dan Dual-Attention

METODE

EfficientNetV2

Merupakan arsitektur jaringan konvolusi (CNN) yang dirancang untuk meningkatkan efisiensi parameter dan mempercepat waktu pelatihan, dengan menggunakan teknik inovatif seperti Neural Architecture Search dan Compound Scaling untuk mencapai keseimbangan optimal antara ukuran parameter dan akurasi.

METODE

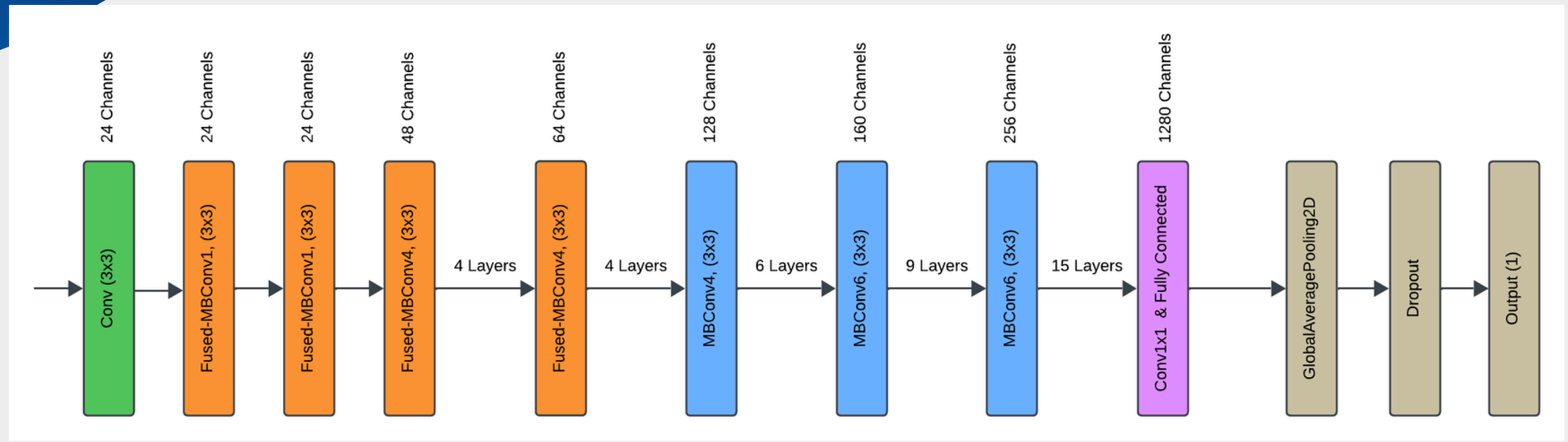
EfficientNetV2 + Attention

Attention atau Squeeze-and-Excitation (SE) adalah mekanisme yang ditambahkan pada CNN untuk memprioritaskan fitur-fitur penting dari input data. Fokusnya adalah pada channel-wise attention, di mana perhatian diberikan pada pentingnya setiap saluran (channel) fitur.

EfficientNetV2 + Dual Attention

Dual-Attention atau Convolutional Block Attention Module (CBAM) adalah mekanisme attention yang lebih canggih, yang memperhatikan dua aspek penting dalam representasi fitur, yaitu channel attention dan spatial attention. Ini membantu model menentukan apa dan di mana fitur penting berada.

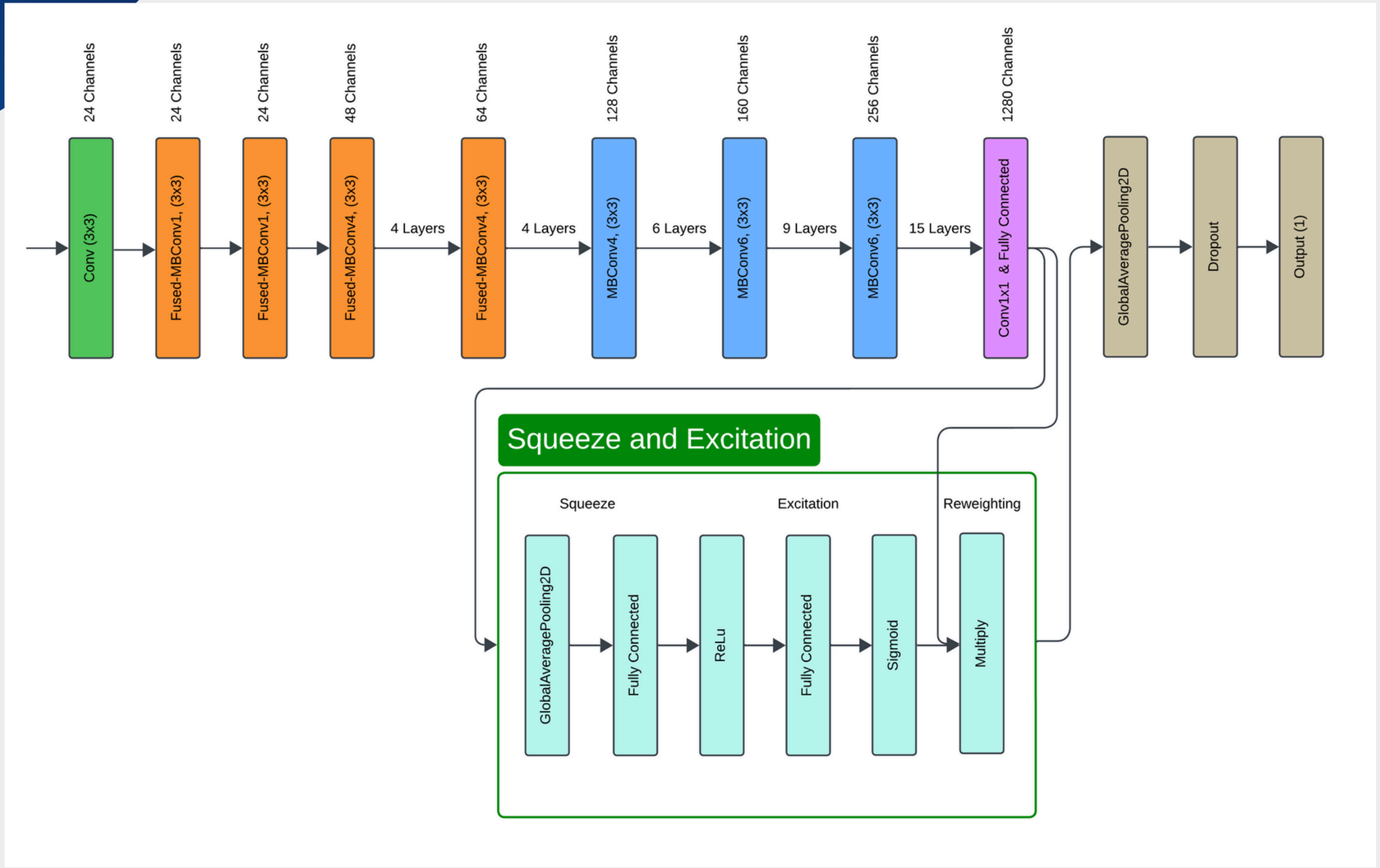
EfficientNetV2



EfficientNetV2

```
# Load model EfficientNetV2
base_model = EfficientNetV2B2(
  weights='imagenet',
  include_top=False,
  input_shape=(img_height, img_width, 3)
)

# Bekukan lapisan base_model
base_model.trainable = False

# Tambahkan lapisan atas
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.2)(x)
outputs = Dense(1, activation='sigmoid')(x)

# Buat model lengkap
model = Model(inputs=base_model.input, outputs=outputs)
```

Single Attention (Squeeze-and-Excitation)

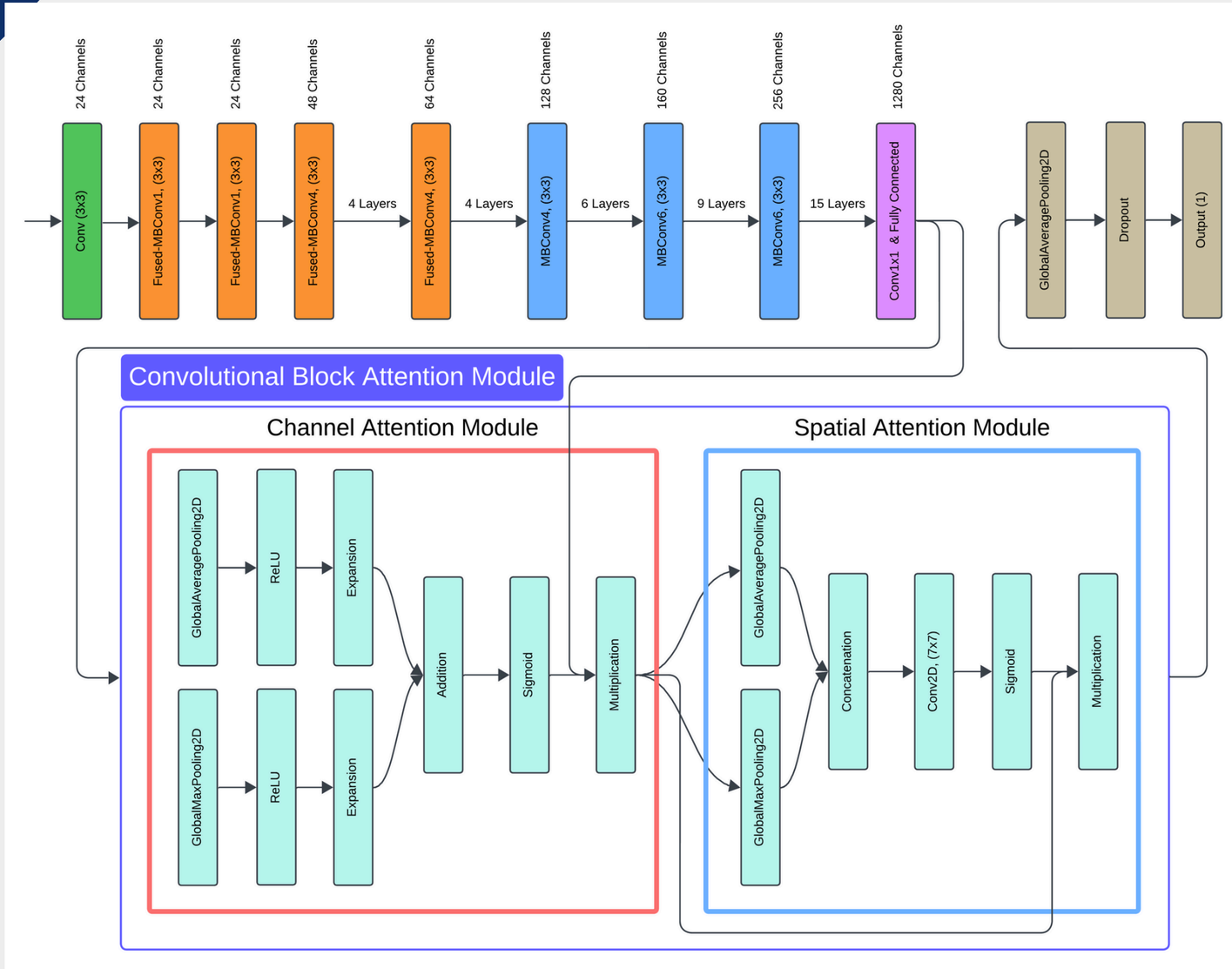


Single Attention (Squeeze-and-Excitation)

```
# Definisikan fungsi SE Block (Squeeze-and-Excitation)
def se_block(input_tensor, ratio=16):
  '''Create a squeeze-and-excitation block'''
  channel_axis = -1
  filters = input_tensor.shape[channel_axis]
  se_shape = (1, 1, filters)

  # Squeeze
  se = GlobalAveragePooling2D()(input_tensor)
  se = Reshape(se_shape)(se)

  # Excitation
  se = Dense(filters // ratio, activation='relu')(se)
  se = Dense(filters, activation='sigmoid')(se)

  # Reweighting
  x = Multiply()([input_tensor, se])
  return x
```

```
# Tambahkan lapisan atas dengan SE Block
x = base_model.output
x = se_block(x) # Tambah SE Block
x = GlobalAveragePooling2D()(x)
x = Dropout(0.2)(x)
outputs = Dense(1, activation='sigmoid')(x)

# Buat model lengkap
model = Model(inputs=base_model.input,
outputs=outputs)
```

Dual Attention (Convolutional Block Attention Module)



Dual Attention (Convolutional Block Attention Module)

```
# Definisikan fungsi CBAM (Convolutional Block Attention Module)
def cbam_block(input_feature, ratio=8):
  # Channel Attention Module
  channel = input_feature.shape[-1]

  shared_layer_one = Dense(channel//ratio,
             activation='relu',
             kernel_initializer='he_normal',
             use_bias=True,
             bias_initializer='zeros')
  shared_layer_two = Dense(channel,
             kernel_initializer='he_normal',
             use_bias=True,
             bias_initializer='zeros')

  avg_pool = GlobalAveragePooling2D()(input_feature)
  avg_pool = Reshape((1,1,channel))(avg_pool)
  avg_pool = shared_layer_one(avg_pool)
  avg_pool = shared_layer_two(avg_pool)

  max_pool = GlobalMaxPooling2D()(input_feature)
  max_pool = Reshape((1,1,channel))(max_pool)
  max_pool = shared_layer_one(max_pool)
  max_pool = shared_layer_two(max_pool)

  channel_attention = Add()([avg_pool, max_pool])
  channel_attention = Activation('sigmoid')(channel_attention)

  # Apply Channel Attention
  channel_refined_feature = Multiply()([input_feature, channel_attention])
```

```
# Spatial Attention Module
  avg_pool = Lambda(lambda x: tf.reduce_mean(x,
axis=-1, keepdims=True))(channel_refined_feature)
  max_pool = Lambda(lambda x: tf.reduce_max(x, axis=-1,
keepdims=True))(channel_refined_feature)
  concat = Concatenate(axis=-1)([avg_pool, max_pool])
  spatial_attention = Conv2D(filters=1,
                kernel_size=7,
                strides=1,
                padding='same',
                activation='sigmoid',
                kernel_initializer='he_normal',
                use_bias=False)(concat)

  # Apply Spatial Attention
  refined_feature = Multiply()
([channel_refined_feature, spatial_attention])

  return refined_feature
```

Dual Attention (Convolutional Block Attention Module)

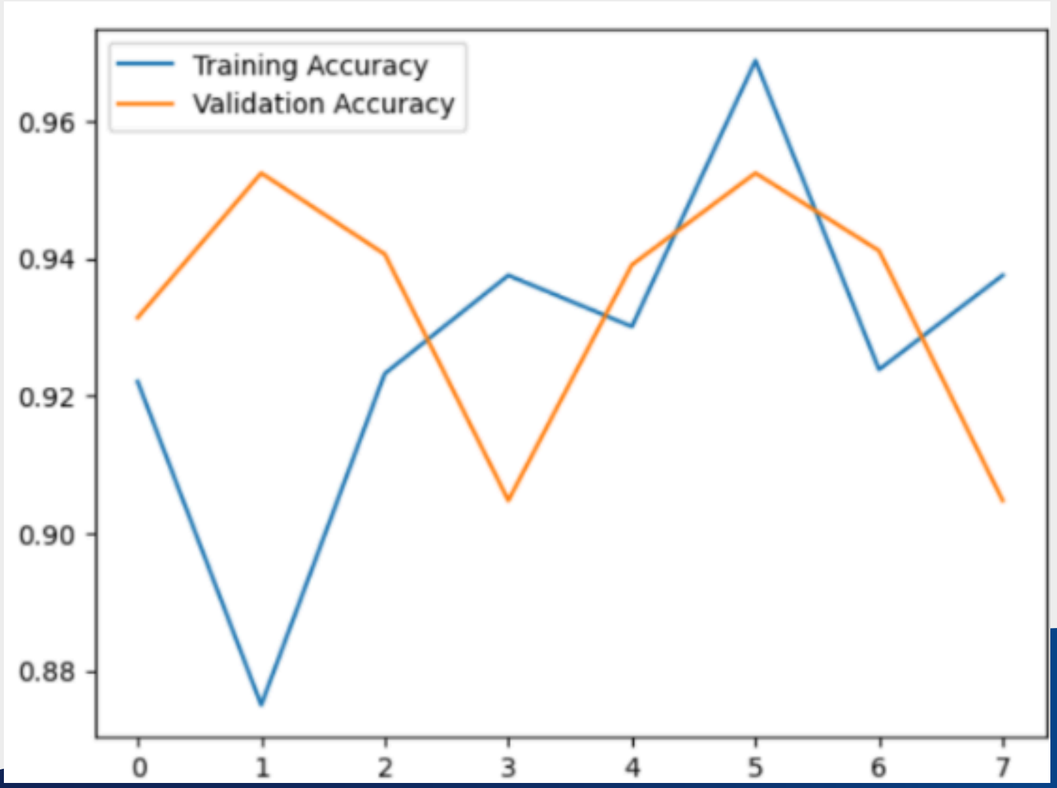
```
# Tambahkan lapisan atas dengan CBAM
x = base_model.output
x = cbam_block(x) # Tambahkan CBAM
x = GlobalAveragePooling2D()(x)
x = Dropout(0.2)(x)
outputs = Dense(1, activation='sigmoid')(x)

# Buat model lengkap
model = Model(inputs=base_model.input,
outputs=outputs)
```

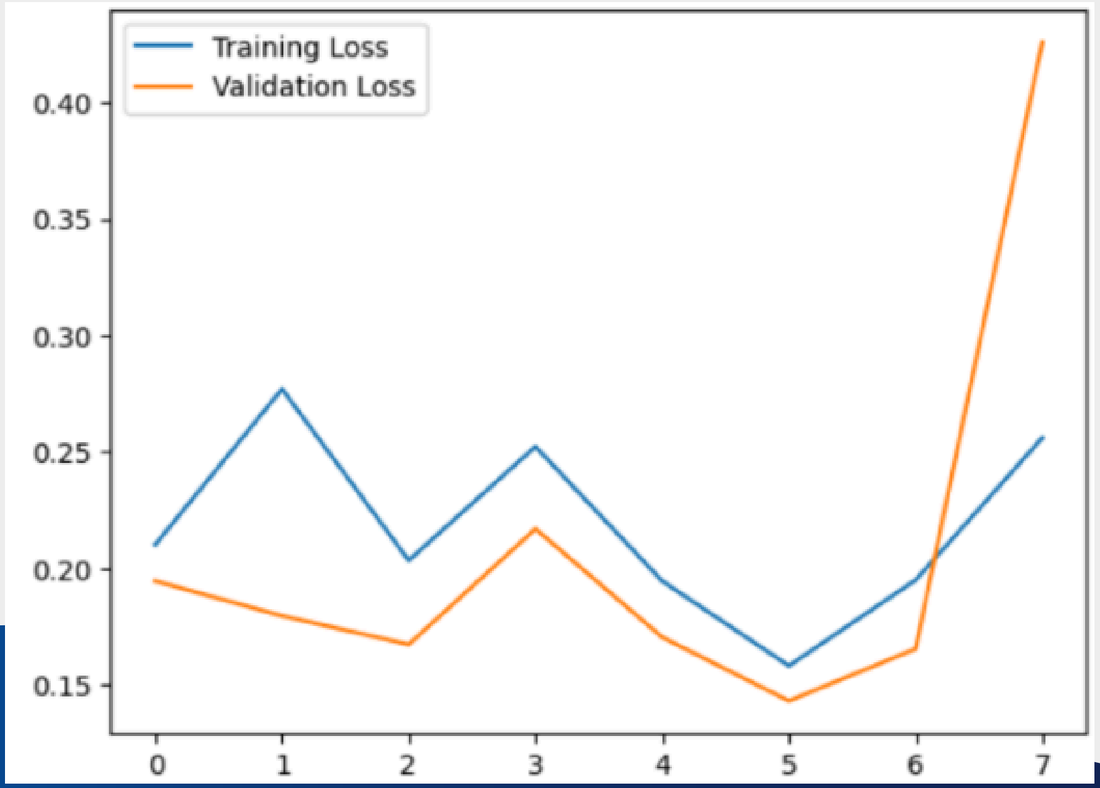
Dataset Percobaan

Label	Train	Valid	Test	Total
Open	3453	986	494	4933
Closed	3455	987	494	4936
Jumlah	6908	1973	988	9869

Hasil Epoch EfficientNetV2

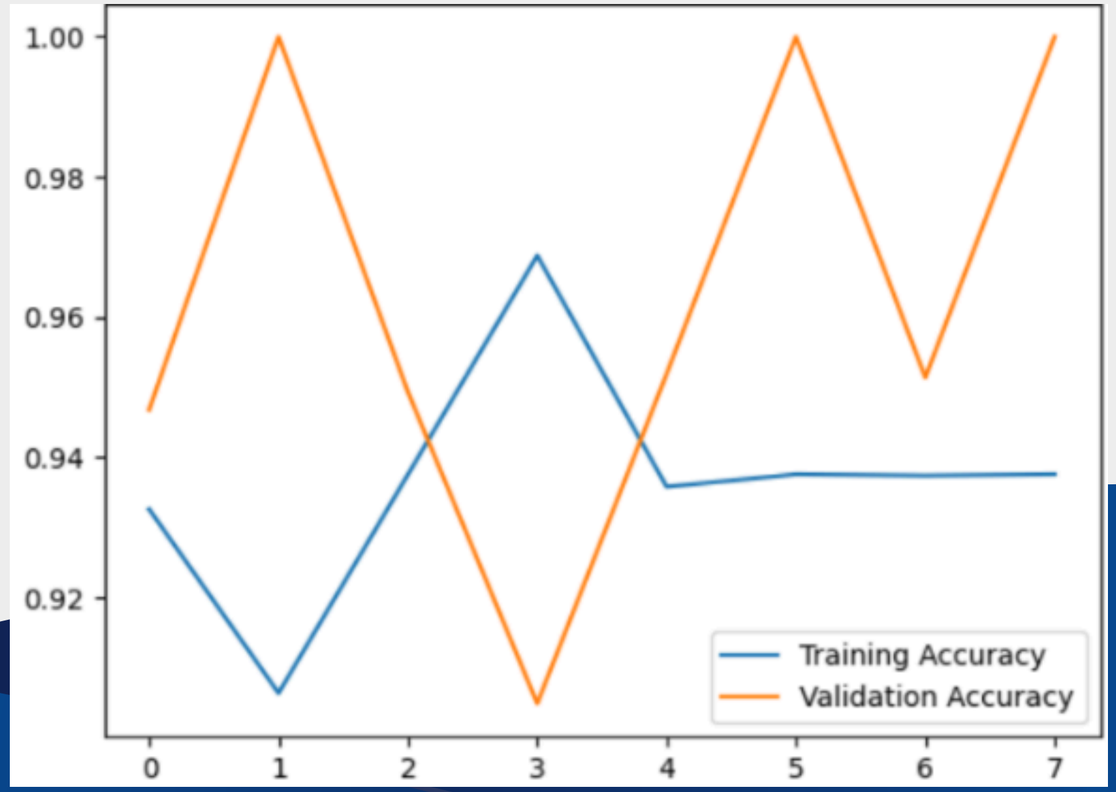
```
Epoch 6/20
215/215 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 301us/step -
accuracy: 0.9688 - loss: 0.1580 - val_accuracy: 0.9524 -
val_loss: 0.1429
```

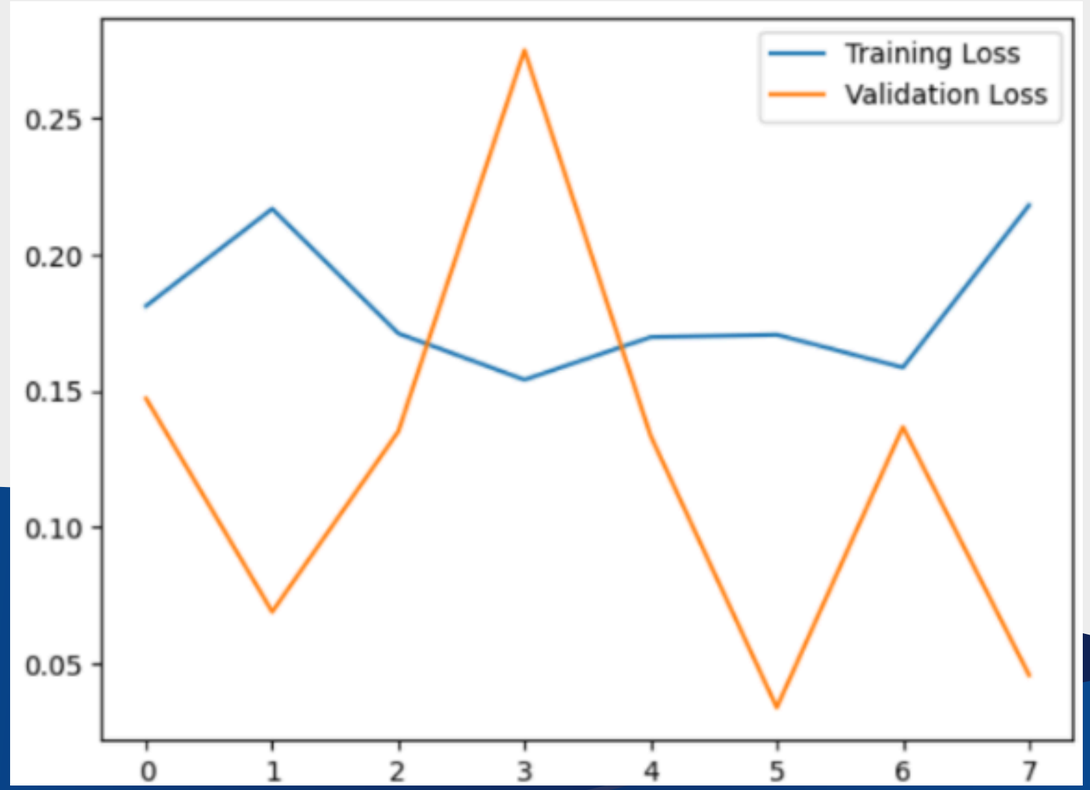




Hasil Epoch EfficientNetV2 + Attention

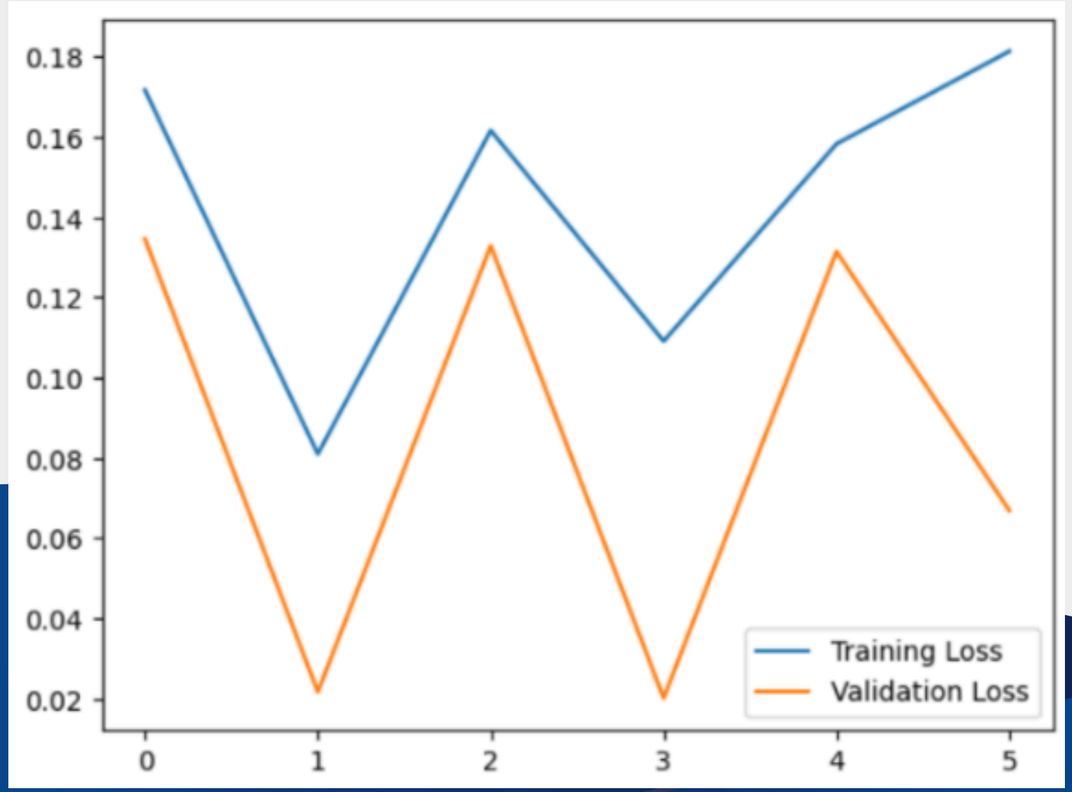
```
Epoch 6/20
215/215 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 21ms/step -
accuracy: 0.9375 - loss: 0.1706 - val_accuracy: 1.0000 -
val_loss: 0.0340
```



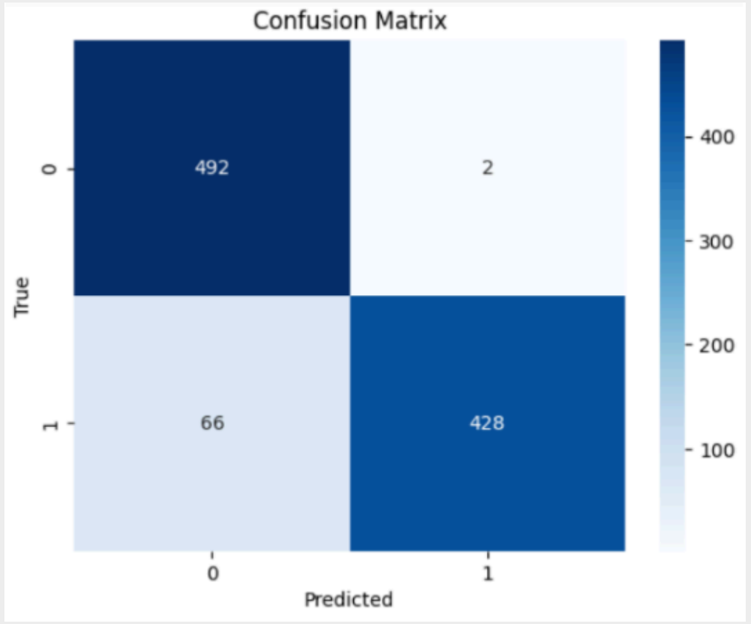


Hasil Epoch EfficientNetV2 + Dual-Attention

```
Epoch 4/20
215/215 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 396us/step -
accuracy: 0.9688 - loss: 0.1091 - val_accuracy: 1.0000 -
val_loss: 0.0201
```

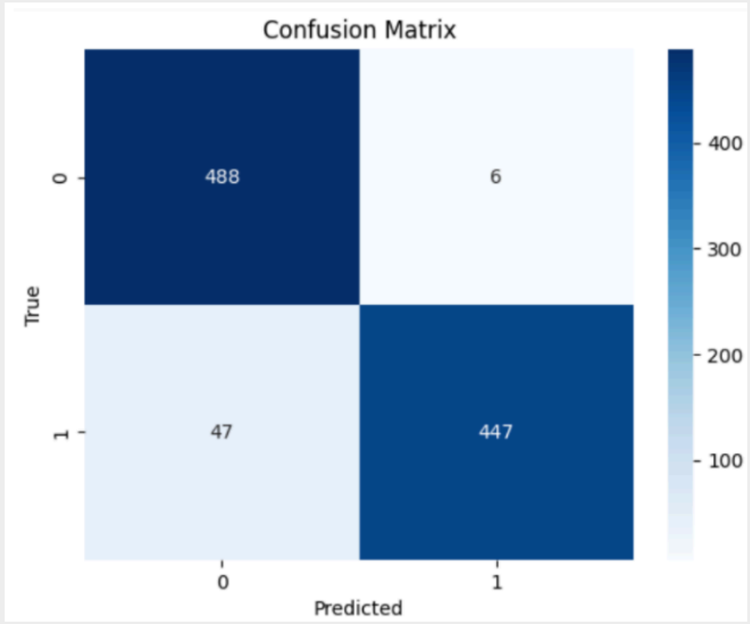


Confusion Matrix



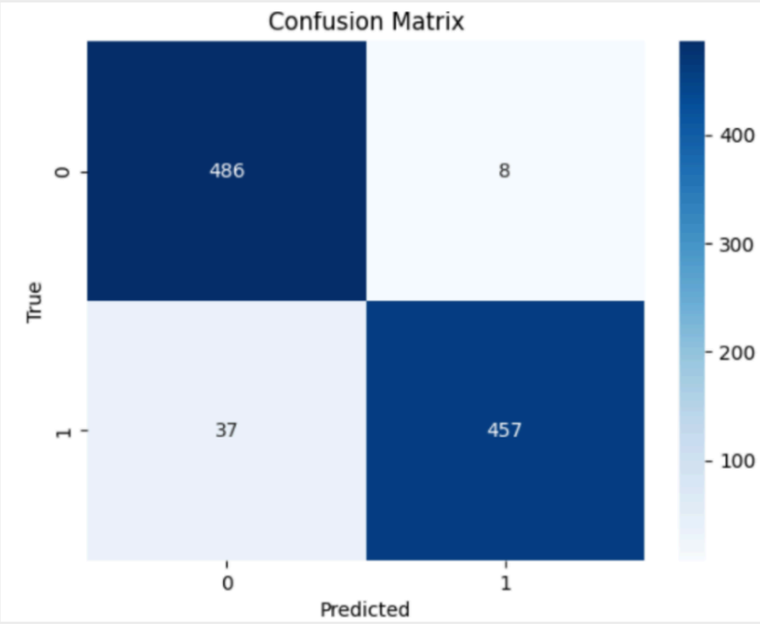
	Precision	Recall	F1-score	Support
Closed	0.88	1.00	0.94	494
Open	1.00	0.87	0.93	494
Accuracy			0.93	988
Macro Avg	0.94	0.93	0.93	988
Weighted Avg	0.94	0.93	0.93	988

EfficientNetV2



	Precision	Recall	F1-score	Support
Closed	0.91	0.99	0.95	494
Open	0.99	0.90	0.94	494
Accuracy			0.95	988
Macro Avg	0.95	0.95	0.95	988
Weighted Avg	0.95	0.95	0.95	988

EfficientNetV2+ Attention



	Precision	Recall	F1-score	Support
Closed	0.93	0.98	0.96	494
Open	0.98	0.93	0.95	494
Accuracy			0.95	988
Macro Avg	0.96	0.95	0.95	988
Weighted Avg	0.96	0.95	0.95	988

EfficientNetV2+ Dual Attention

Jumlah Parameter dan mAP

	Parameter	Size
Total params	8.770.783	33,46 MB
Trainable params	1.409	5,50 KB
Non-trainable params	8.769.374	33,45 MB

```
print(f"Average precision score: {average_precision:.4f}")
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 2s 79ms/step
Average precision score: 0.9869
```

EfficientNetV2

	Parameter	Size
Total params	9.268.081	35,35 MB
Trainable params	98.707	1,90 MB
Non-trainable params	8.769.374	33,45 MB

```
print(f"Average precision score: {average_precision:.4f}")
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 96ms/step
Average precision score: 0.9906
```

EfficientNetV2+ Dual Attention

	Parameter	Size
Total params	9.020.087	34,41 MB
Trainable params	250.713	979,35 KB
Non-trainable params	8.769.374	33,45 MB

```
print(f"Average precision score: {average_precision:.4f}")
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 93ms/step
Average precision score: 0.9900
```

EfficientNetV2+ Attention

Waktu Komputasi

Keterangan	Training	Test
EfficientNetV2	524,14 s, (Avg: 130, 50 s)	2,78 s
EfficientNetV2 + Attention	477,63 s, (Avg: 123, 00 s)	3,73 s
EfficientNetV2 + Dual-Attention	374,47 s, (Avg: 115, 25 s)	3,18 s

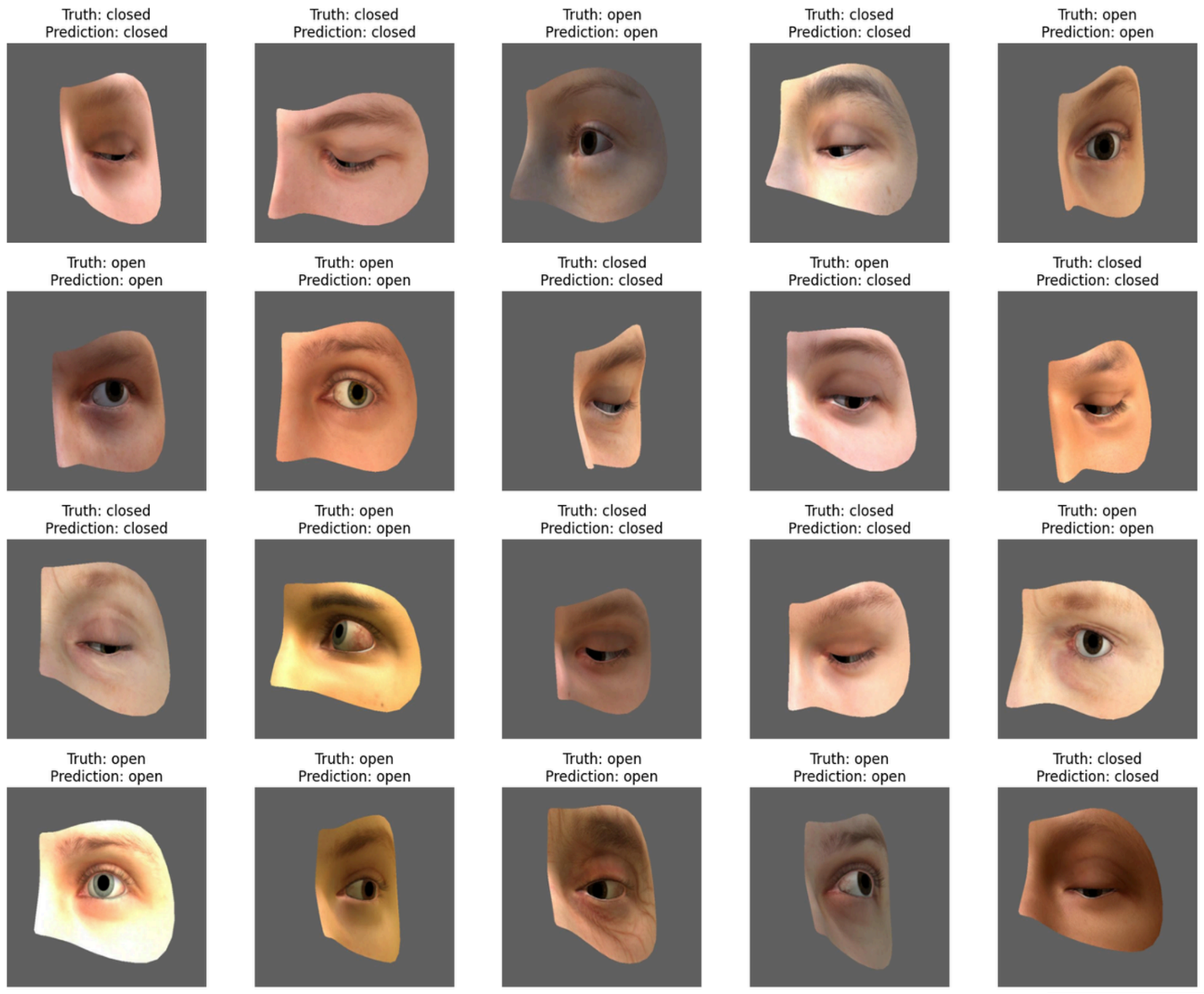
Spesifikasi (Colab):

1.Operating System: Linux 6.1.85+
2.CPU: Inte(R) Xeon(R) CPU @ 2.00GHz
3.RAM: 12.67 GB
4.GPU: Tesla T4
5.GPU Memory Total: 15360.0 MB
6.Python version: 3.10.12

Tabel Perbandingan Metode

Keterangan	EfficientNetV2	EfficientNetV2 + *Attention*	EfficientNetV2 + *Dual-Attention*
Total Parameter	8.770.783	9.020.087	9.268.081
Akurasi Terbaik	0.9524	1.0000	1.0000
Loss (Val)	0.1429	0.0340	0.0201
mAP	0.9869	0.9900	0.9906
Waktu *Train* (avg)	130,50 s	115,25 s	123.00 s
Waktu *Test*	2,78 s	3,73 s	3,18 s
Precision	0,8663/0,9959	0,9878/ 0,9878	0,9292/0,9827
Recall	0,8663/0,9959	0,9878/0,9048	0,9838/0,9251
F1-*Score*	0,9353/0,9264	0,9336/0,9440	0,9838/0,9251

Pengujian EfficientNetV2



Pengujian EfficientNetV2 + Attention

Pengujian EfficientNetV2 + Dual-Attention

Kesimpulan

Dengan akurasi dari pengujian model, dapat dikatakan penggunaan attention dan dual-attention dapat memberikan peningkatan kemampuan akurasi pengujian model. Penggunaan attention dan dual-attention juga memberikan peningkatan berbagai hal lain, seperti yang ditunjukkan pada tabel sebelumnya. Dengan demikian, dapat disimpulkan bahwa penggunaan attention dan dual-attention pada model EfficientNetV2 berhasil meningkatkan kemampuan dari model tersebut.

THANK YOU